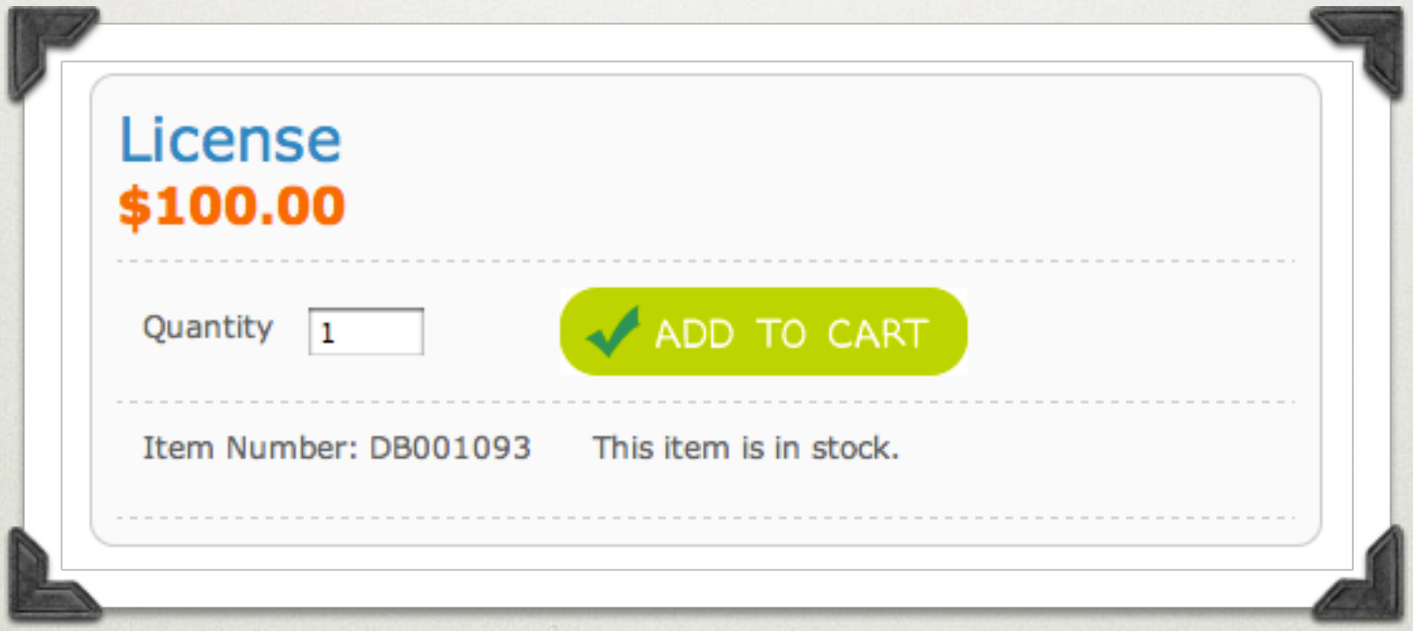
Internet Programming in Python - Week 10

The Cloud & App Engine



http://thedilbertstore.com/comic_strips/2011/01/07/products?utm_source=dilbert.com&utm_medium=button&utm_campaign=License%2BThis%20Strip&refcode=DB1020#licensing-header

Brian Dorsey
brian@dorseys.org

A moment to reflect

Japan

ANNOUNCEMENTS

-

Blue Box thank you card

(or “thank you paper”?)

“thank you poster” ?

next quarter

Questions and Review (20)

some thoughts from the assignments

Assignment

turn in here: `http://is.gd/uwipip_week9`

- Sign up for Google App Engine:
  `https://appengine.google.com/` (phone with SMS needed)
- List all of the queries in your week 7/8 app
- Import your week 7/8 data into a NOSQL database. Maybe App Engine or MongoDB?
- Attempt to write equivalent queries against your NOSQL DB.
- Check in your import script and query script.
- stretch - port your week 7/8 app to use the NOSQL DB.
- super stretch - port your week 7/8 app to App Engine using django-nonrel:
  `http://www.allbuttonspressed.com/projects/django-nonrel`

Would you rather do something different with NOSQL? Ask me, it's probably OK.

some examples

MongoDB commands

```
db.food.save({ name: 'Beef Stew', category:
'Dinner', protein: 5, net_carbs: 22, ingreds:
[{name: 'onions', amount : '1 Cup'}]})
...
db.food.save({ name: 'Cottage Cheese Pancakes',
category: 'Breakfast', protein: 5, net_carbs: 6,
ingreds: [{name: 'cottage cheese', amount : '3
Cups'}]})

##########################

db.food.find({'category': 'Dinner'})
db.food.find({}).sort({name: 1});
```

mongo in python

```
from pymongo import Connection
connection = Connection()

db = connection['rpg-database']

rpg0 = {'name': 'Advanced Dungeons and Dragons',
        'year': '2010',
        'publisher': 'Wizards of the Coast',
        'medium': 'Pen and paper',
        'description': 'Enter a magical world of wizardry and heroism!',
       }
<.............  repeat  ........................>
rpg5 = {'name': "Baldur's Gate",
        'year': '1999',
        'publisher': 'Bioware',
        'medium': 'PC',
        'description': 'Using Dungeons and Dragons ruleset, this RPG set a
number of milestones on the PC.',
       }

rpgs = [rpg0, rpg1, rpg2, rpg3, rpg4, rpg5]
post = db.post

for rpg in rpgs:
    post.insert(rpg)
```

sqlite --> mongo

```
import pymongo, sqlite3

sqlite_conn = sqlite3.connect('sqlite3.db')
c = sqlite_conn.cursor()

mongo_conn = pymongo.Connection("localhost", 27017)
family_db = mongo_conn.family

c.execute('SELECT * FROM "family_family" ORDER BY
"family_family"."id"')
for family in c:
   family_db.families.save({'_id': family[0],
'name': family[1],
      'pub_date': family[2]})
```

sqlite --> mongo (2)

```
# Connect to the Mongo DB
connection = Connection('localhost', 27017)
db = connection.book_db
collection = db.book_db
posts = db.posts

def move() :
    """This method extracts the data from the sqlite3 database and stores it
in the google data store"""

    global posts

    conn = sqlite3.connect('/home/jeffs/python/PythonClass/week7/mysite/bookdb/sqlite3.db')
    c = conn.cursor()
    c.execute('select * from bookdb_book')

    for row in c:
        ident, title, copyright_date, author, isbn, location = row
        post = { "ident": ident,
                 "title": title,
                 "copyright_date": copyright_date,
                 "author": author,
                 "isbn": isbn,
                 "location": location,
                }
        posts.insert(post)
    conn.close()
```

mongo queries (2)

```
def check_by_title() :
    """List the database by title"""
    for row in posts.find().sort("title") :
        print row["ident"], row["title"], row
["copyright_date"], row["author"], row["isbn"], row
["location"]
```

```
def check_by_location() :
    """List the database by location"""
    for row in posts.find().sort("location") :
        print row["ident"], row["title"], row
["copyright_date"], row["author"], row["isbn"], row
["location"]
```

postgres to gae datastore

```
#!/bin/bash
rm games_output.csv

psql -d boardgames -F ',' -A -o games_output.csv --quiet -c "select id,title,
publisher,year_published, description, to_char(last_played, 'YYYY-MM-DD hh:mm:ss')
as last_played, image_name, genre, maxplayers, minplayers from gameviewer_game;"

# remove the last line from the file
sed -ie '$d' games_output.csv

~/opt/google_appengine/appcfg.py upload_data --config_file=bulkloader.yaml --
filename=games_output.csv --kind=gameviewer_game --url=http://`hostname`:8000/
remote_api --application=ben-cleveland
#~/opt/google_appengine/appcfg.py upload_data --config_file=bulkloader.yaml --
filename=games_output.csv --kind=gameviewer_game  --application=ben-cleveland ./

rm rating_output.csv
psql -H -d boardgames -F $'\t' -A -o rating_output.csv --quiet -c "select id*10 as
id, game_id, rating, comment, name from gameviewer_rating;"

# remove the last line from the file
sed -ie '$d' rating_output.csv

~/opt/google_appengine/appcfg.py upload_data --config_file=bulkloader.yaml --
filename=rating_output.csv --kind=gameviewer_rating --url=http://`hostname`:8000/
remote_api --application=ben-cleveland
```

Lecture A (25)

0:20

the cloud

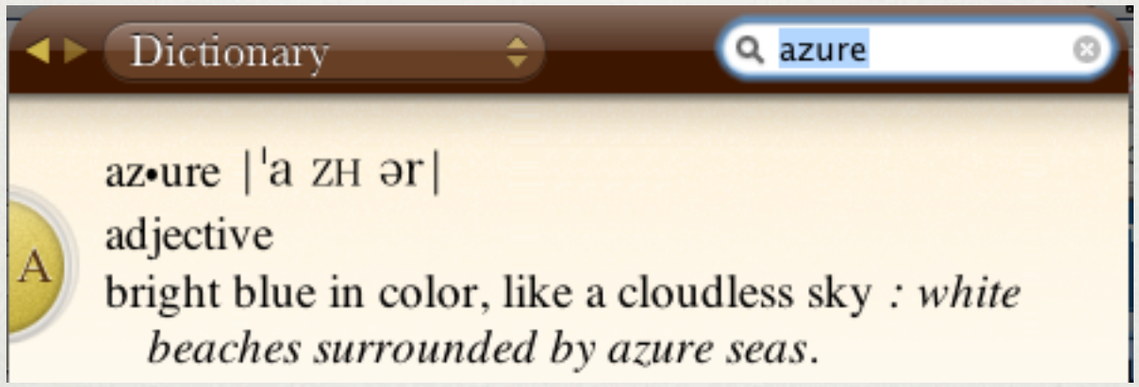
definition?

amazon aws

rackspace

any service which offers VMs and an API (libcloud)

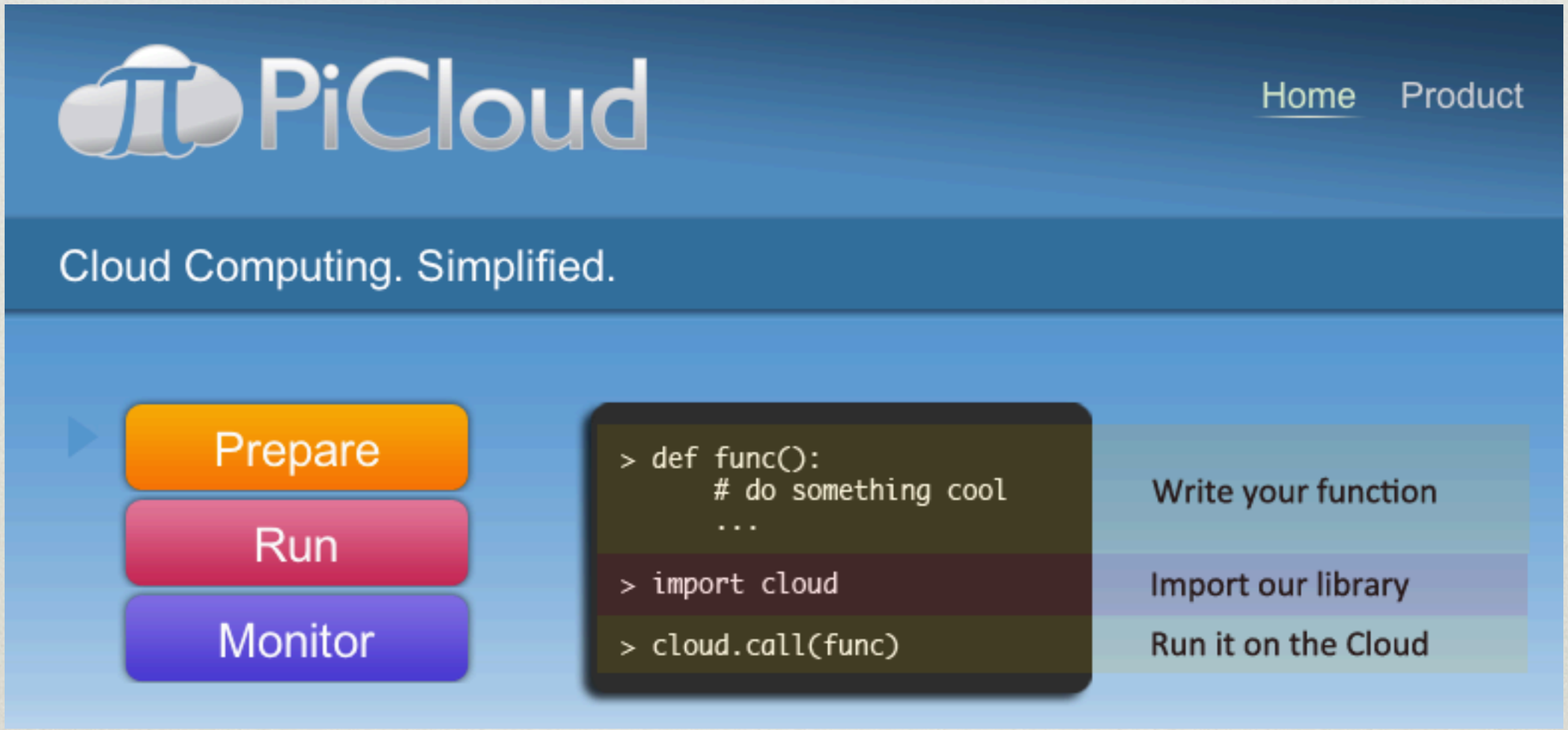
MS Azure


Dictionary
azure
az•ure |ˈa ZH ər|
adjective
A
bright blue in color, like a cloudless sky : *white beaches surrounded by azure seas.*

Really??!?!!?!

heroku

PiCloud



http://www.picloud.com/

app engine

app engine

Google App Engine == GAE

a magic CGI host

WSGI adapter as well

tradeoff:

locked down environment
for
no sysadmin

no filesystem

no installing C extensions

no remote services except those specifically offered by GAE

use HTTP to connect to the outside world (in and out)

at a high level, what they're really offering is automatic process management and the datastore

good

magic scaling goodness

free for small stuff

no admin

bad

unpredictable latency, both from spin up, and datastore

no real recourse when things go wrong

(no admin)

quotas

summary from:

http://en.wikipedia.org/wiki/Google_App_Engine

details here:

http://code.google.com/appengine/docs/quotas.html

basically, it's free for any hobby level projects

domains

custom domain support via Google Apps for your Domain

http://www.google.com/apps/intl/en/group/index.html

or yourapp.appspot.com - unique name across ALL GAE users!

development cycle

SDK

local dev server, with emulated environment

app.yaml

http://code.google.com/appengine/docs/python/config/appconfig.html

appcfg.py

```
Usage: appcfg.py [options] <action>

Action must be one of:
  create_bulkloader_config: Create a bulkloader.yaml from a
running application.
  cron_info: Display information about cron jobs.
  download_app: Download a previously-uploaded app.
  download_data: Download entities from datastore.
  help: Print help for a specific action.
  request_logs: Write request logs in Apache common log format.
  rollback: Rollback an in-progress update.
  set_default_version: Set the default (serving) version.
  update: Create or update an app version.
  update_cron: Update application cron definitions.
  update_dos: Update application dos definitions.
  update_indexes: Update application indexes.
  update_queues: Update application task queue definitions.
  upload_data: Upload data records to datastore.
  vacuum_indexes: Delete unused indexes from application.
Use 'help <action>' for a detailed description.
```

static files

http://code.google.com/appengine/docs/python/gettingstarted/staticfiles.html

push to GAE

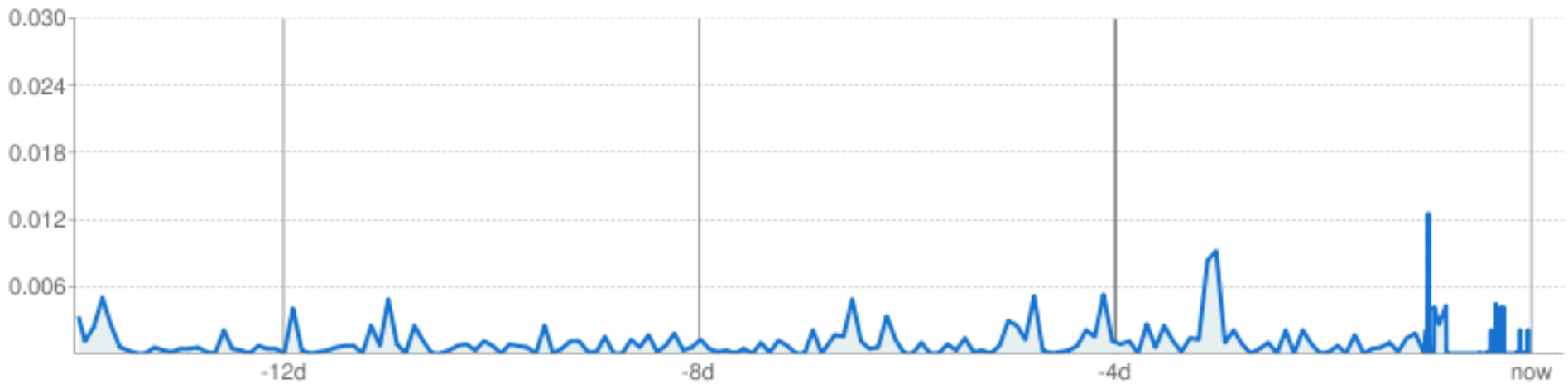
multiple versions running

same datastore

admin UI - datastore queries, logs, etc

Charts

Requests/Second | 6 hrs | 12 hrs | 24 hrs | 2 days | 4 days | 7 days | 14 days | 30 days



Instances

Number of Instances - Details	Average QPS	Average Latency	Average Memory
0 total	Unknown	Unknown ms	Unknown MBytes

Billing Status: Enabled (Daily budget: $2.00) - Settings

Quotas reset every 24 hours. Next reset

Resource		Usage			Cost / Budget
CPU Time	$0.10/CPU hour		0%	0.00 of 16.60 CPU hours	$0.00 / $1.01
Outgoing Bandwidth	$0.12/GByte		0%	0.00 of 5.17 GBytes	$0.00 / $0.50
Incoming Bandwidth	$0.10/GByte		0%	0.00 of 2.60 GBytes	$0.00 / $0.16
Total Stored Data	$0.005/GByte-day		0%	0.28 of 67.00 GBytes	$0.00 / $0.33
Recipients Emailed	$0.0001/Email		0%	0 of 2,000	$0.00 / $0.00
		= Free quota		Estimated cost for the last 13 hours:	**$0.00 / $2.00**

Current Load

URI	Requests last 13 hrs	Avg CPU (API) last hr	% CPU last 13 hrs
/	6	447 (0)	17%
/api/1/update/	2	0 (0)	83%

Errors

URI	Count

status

http://code.google.com/status/appengine/

Google app engine

Help | Admin Console

System Status

Python runtime issues – Python
Mar 08 2011, 03:00 AM - Mar 08 2011, 08:00 AM
posted by mickey

The Python runtime was affected by a bad push to production this morning. Service has been restored to some lingering applications now and we are working on a post-mortem describing what went wrong.

The Downtime Notification for this outage is here:
http://groups.google.com/group/google-appengine-downtime-notify/browse_thread/thread/95c487775d6a748d

Current Availability

100%

Uptime (last 7 days)

Read latency (today)

Write latency (today)

	03/08/11	03/09/11	03/10/11	03/11/11	03/12/11	03/13/11	Yesterday	Today	Now
Serving									
Python	⚠	✓	✓	✓	✓	✓	✓	✓	Normal
Java	✓	✓	✓	✓	✓	✓	✓	✓	Normal
APIs									
Datastore	✓	✓	✓	✓	✓	✓	✓	✓	Normal
HR Datastore	✓	✓	✓	✓	✓	✓	✓	✓	Normal
Images	✓	✓	✓	✓	✓	✓	✓	✓	Normal
Mail	✓	✓	✓	✓	✓	✓	✓	✓	Normal
Memcache	✓	✓	✓	✓	✓	✓	✓	✓	Normal
Taskqueue	✓	✓	✓	✓	✓	✓	✓	✓	Normal
Urlfetch	✓	✓	✓	✓	✓	✓	✓	✓	Normal
Users	✓	✓	✓	✓	✓	✓	✓	✓	Normal

people with experience say that individual application downtime isn't represented here, *only* large scale outages

frameworks: django

django

webapp

http://code.google.com/appengine/docs/python/tools/webapp/

web2py

http://www.web2py.com/

“**tipfy** is a small but powerful framework made specifically for Google App Engine. It is a lot like webapp ... but offers a bunch of features and goodies that webapp misses: i18n, sessions, own authentication, flash messages and more. Everything in a modular, lightweight way, tuned for App Engine. You use only what you need, when you need.”

http://www.tipfy.org/

Break

(10)

0:45

Lab A

(20)

0:55

Lab A

- write a “hello world” app and upload it
- `http://code.google.com/appengine/docs/python/gettingstarted/helloworld.html`

  `http://code.google.com/appengine/docs/python/gettingstarted/uploading.html`
- stretch: print all environment variables:
  `http://code.google.com/appengine/docs/python/runtime.html#The_Environment`
- stretch: add logging, and view in admin
  `http://code.google.com/appengine/docs/python/runtime.html#Logging`

Lecture B
(15)

1:15

services

http://code.google.com/appengine/docs/python/apis.html

url fetch

make http requests

`http://code.google.com/appengine/docs/python/urlfetch/overview.html`

memcache

RAM based cache

http://code.google.com/appengine/docs/python/memcache/overview.html

blobstore

(kinda like amazon S3)

http://code.google.com/appengine/docs/python/blobstore/

task scheduler

(cron)

http://code.google.com/appengine/docs/python/config/cron.html

task queue

asynchronous task execution

http://code.google.com/appengine/docs/python/taskqueue/overview.html

xmpp

chat & two-way messaging

http://code.google.com/appengine/docs/python/xmpp/overview.html

channel api

one way

persistent connection

http://code.google.com/appengine/docs/python/channel/overview.html

mail

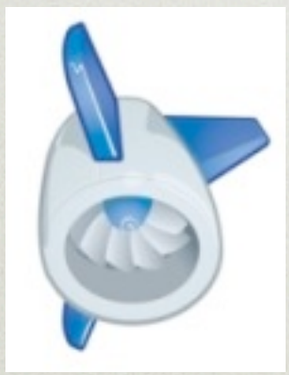
send mail function

incoming mail http requests

http://code.google.com/appengine/docs/python/mail/overview.html




image manipulation



http://code.google.com/appengine/docs/python/images/overview.html

multitenancy

subdivide datastore, memcache, and tasks

http://code.google.com/appengine/docs/python/multitenancy/

tools

http://code.google.com/appengine/tools_tips.html

appstats

detailed runtime statistics

http://code.google.com/appengine/docs/python/tools/appstats.html

AppStats - appoverflow

http://1.latest.appoverflow.appspot.com/stats/?

Google app engine

Application Stats for appoverflow

Refresh Now

RPC Stats	Expand All
RPC	**Count**
datastore_v3.Get	5
datastore_v3.RunQuery	4
user.CreateLoginURL	4
user.CreateLogoutURL	4
datastore_v3.BeginTransaction	1
datastore_v3.Commit	1
datastore_v3.Next	1
datastore_v3.Put	1

Path Stats			Expand All
Path	**#RPCs**	**#Requests**	**Most Recent requests**
/questions/1/	12	3	(2) (5) (7)
POST /questions/1/	5	1	(3)
/	4	1	(8)
/favicon.ico	0	3	(1) (4) (6)

Requests History Expand All

Request

(1) 2010-03-23 07:13:17.867 "GET /favicon.ico" 404 real=0ms cpu=10ms api=0ms overhead=0ms (0 RPCs)

(2) 2010-03-23 07:13:17.311 "GET /questions/1/" 200 real=52ms cpu=56ms api=57ms overhead=0ms (4 RPCs)

(3) 2010-03-23 07:13:16.946 "POST /questions/1/" 302 real=222ms cpu=69ms api=150ms overhead=1ms (5 RPCs)

(4) 2010-03-23 07:13:01.714 "GET /favicon.ico" 404 real=0ms cpu=7ms api=0ms overhead=0ms (0 RPCs)

(5) 2010-03-23 07:12:54.616 "GET /questions/1/" 200 real=63ms cpu=57ms api=48ms overhead=1ms (4 RPCs)

(6) 2010-03-23 07:12:43.219 "GET /favicon.ico" 404 real=0ms cpu=6ms api=0ms overhead=0ms (0 RPCs)

(7) 2010-03-23 07:12:42.947 "GET /questions/1/" 200 real=59ms cpu=82ms api=48ms overhead=0ms (4 RPCs)

(8) 2010-03-23 07:12:04.365 "GET /" 200 real=203ms cpu=1340ms api=214ms overhead=0ms (4 RPCs)

http://googleappengine.blogspot.com/2010/03/easy-performance-profiling-with.html

AppStats - appoverflow

http://1.latest.appoverflow.appspot.com/stats/details?time=1269358561796

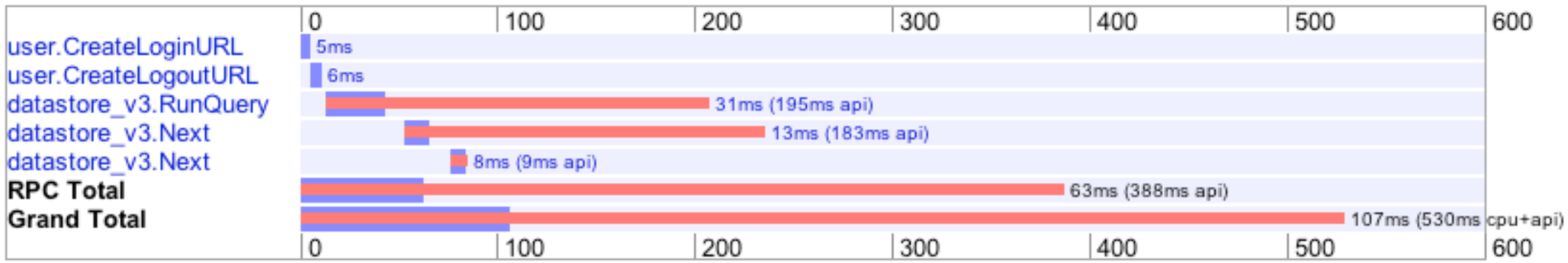
Google app engine

Application Stats for appoverflow

2010-03-23 07:36:01.796 GET /

200 [redacted]@google.com* real=107ms cpu=141ms api=388ms overhead=1ms

Timeline | RPC Stats | CGI Environment | sys.path



RPC Call Traces Expand All

RPC
@0ms **user.CreateLoginURL** real=5ms api=0ms
@5ms **user.CreateLogoutURL** real=6ms api=0ms
@13ms **datastore_v3.RunQuery** real=31ms api=195ms
@53ms **datastore_v3.Next** real=13ms api=183ms
@76ms **datastore_v3.Next** real=8ms api=9ms

AppStats - appoverflow

http://1.latest.appoverflow.appspot.com/stats/details?time=1269

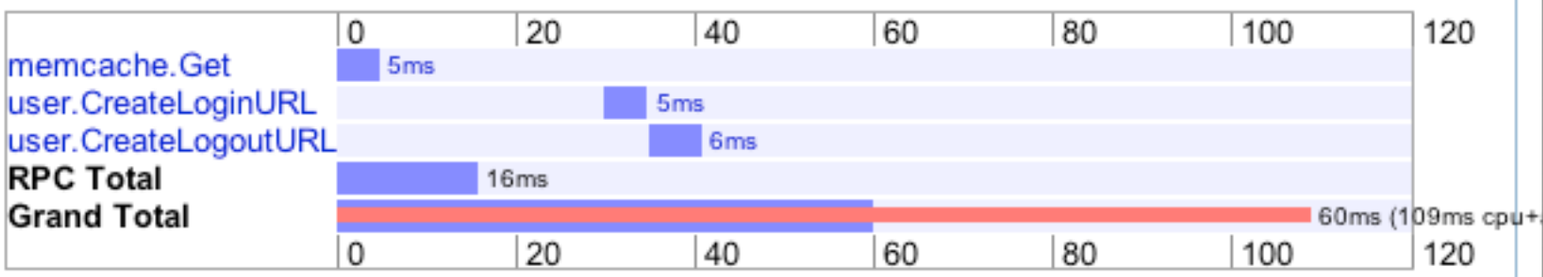
Google app engine

Application Stats for appoverflow

2010-03-30 06:31:10.223 GET /

200 [redacted]@google.com* real=60ms cpu=109ms api=0ms overhead=0ms

Timeline | RPC Stats | CGI Environment | sys.path



RPC Call Traces Expand All

RPC
@0ms **memcache.Get** real=5ms api=0ms
@30ms **user.CreateLoginURL** real=5ms api=0ms
@35ms **user.CreateLogoutURL** real=6ms api=0ms

zipserve

serve static files from a zip file

http://code.google.com/p/googleappengine/source/browse/trunk/python/google/appengine/ext/zipserve/__init__.py

remote_api

use the datastore api from outside of app engine

http://code.google.com/appengine/articles/remote_api.html

bulk load

external data

dump and load

http://code.google.com/appengine/docs/python/tools/uploadingdata.html

django app

as a source of useful modules (simplejson, etc)

http://code.google.com/appengine/docs/python/tools/libraries.html

insulation

are we locked in?

just use cgi/wsgi?

what about the datastore?

frameworks

django

django-nonrel

http://www.allbuttonspressed.com/projects/django-nonrel

http://pycon.blip.tv/file/4881543/

web2py

http://web2py.com/book/default/chapter/11#Google-App-Engine

app engine emulators

TyphoonAE

http://code.google.com/p/typhoonae/

AppScale

http://code.google.com/p/appscale/

articles!

http://code.google.com/appengine/articles/

Lab B

(20)

1:30

Lab B

- Lets do the addition example again!
- Make a url which accepts a= & b= parameters, and displays the sum & the current date
- Then, make a datastore class to store the date and inputs, modify the app to store every pair of inputs and the date requested. View the data in the admin.
- stretch: make a page which lists the last 20 additions, in date order (recent first).
- stretch: compute additional stats on the last 20 additions, min, max, mean, etc.
- double stretch: can you show the total number of addition requests ever?

Break

(10)

1:50

PyCon

(30)

2:00

observations

convore

so many good talks

`http://us.pycon.org/2011/schedule/`

Using Python 3 to Build a Cloud Computing Service for my Superboard II

David Beazley

Javascript for people who know Python

Ian Bicking

Best Practices for Impossible Deadlines

Christopher Groskopf

most videos already online

http://pycon.blip.tv/posts

photos

RANDOM

WRAPUP

2:30

Blue Box VMs

Assignment

- Self evaluation - I'll send a form to the list.
- Think about project ideas for next quarter.
- If group work is possible for your idea, please post it to the list and see if anyone is interested in working on it with you. Also, look at the ideas and see if you'd like to work on them.

Evaluation

2:30